Effective Feature Construction by Maximum Common Subgraph Sampling

Leander Schietgat, Fabrizio Costa, Jan Ramon, and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. The standard approach to feature construction and predictive learning in molecular datasets is to employ computationally expensive graph mining techniques and to bias the feature search exploration using frequency or correlation measures. These features are then typically employed in predictive models that can be constructed using, for example, SVMs or decision trees. We take a different approach: rather than mining for all optimal local patterns, we extract features from the set of pairwise maximum common subgraphs. The maximum common subgraphs are computed under the block-and-bridge-preserving subgraph isomorphism from the outerplanar examples in polynomial time. We empirically observe a significant increase in predictive performance when using maximum common subgraph features instead of correlated local patterns on 60 benchmark datasets from NCI. Moreover, we show that when we randomly sample the pairs of graphs from which to extract the maximum common subgraphs, we obtain a smaller set of features that still allows the same predictive performance as methods that exhaustively enumerate all possible patterns. The sampling strategy turns out to be a very good compromise between a slight decrease in predictive performance (although still remaining comparable with state-of-the-art methods) and a significant runtime reduction (two orders of magnitude on a popular medium size chemoinformatics dataset). This suggests that maximum common subgraphs are interesting and meaningful features.

Keywords: feature generation, subgraph mining, structure-activity learning, chemoinformatics

1 Introduction

During the last decade, a lot of attention has been devoted to mining local patterns in molecular datasets, leading to the development of many graph mining systems. These systems typically employ constraints to specify the patterns of interest, such as frequency, or top-k according to a correlation measure (e.g., χ^2). Graph mining systems then perform a complete search through the entire graph space, enumerating all subgraphs satisfying these constraints [1, 2] or even exhaustively enumerating all possible subgraphs [3].

Usually the resulting patterns are not used directly. Instead, they are used as features in combination with traditional machine learning algorithms. Furthermore, the quality of the generated patterns is measured through the quality of the induced classifiers or models for regression [3]. While these approaches offer strong guarantees w.r.t. completeness or optimality of the found patterns, they have a high computational cost and require post-processing to deal, for example, with redundancy issues [2]. In this way, local pattern mining acts as a complex, expensive and *indirect* approach to generate features for graphs.

We propose a simple, direct and efficient approach to generate interesting graph patterns. The idea is to compute maximum common subgraphs from randomly selected pairs of examples and to directly use them as features. While computing maximum common subgraphs in general is an NP-hard problem, a polynomial-time algorithm exists for outerplanar graphs in combination with the block-and-bridge-preserving subgraph isomorphism [4]. Outerplanar graphs can be embedded in the plane such that all of their vertices lie on the outside of the graph. It is known that 95% of the molecules in the NCI[1] collection are outerplanar [5], which makes this class of graphs well-suited for molecular datasets. Moreover, it has been shown that employing the block-and-bridge-preserving subgraph isomorphism instead of the general subgraph isomorphism in graph miners yields more predictive feature sets for molecular datasets [4].

The present article extends this earlier work in that it shows that extracting maximum common subgraphs of pairs of molecules in this framework yields even better features. The advantages of this approach are 1) that it is easy to control the number of produced features, while setting the frequency in a pattern mining task yields an unpredictable number of patterns; 2) that patterns can be extracted in polynomial time and more efficiently than by frequent or correlated subgraph mining, as no search space has to be traversed; and 3) that on 60 benchmark problems from NCI, the extracted features allow for the construction of SVM classification models that achieve significantly better predictive performance than those built using features returned by traditional local pattern mining and exhaustive fingerprint generation methods.

The text is organized as follows. We start by explaining the algorithm to compute a maximum common subgraph of two outerplanar graphs and the feature construction method based on maximum common subgraphs in Sect. 2. Section 3 presents an experimental evaluation, showing multiple variants of our method and comparing them to the state-of-the-art. In Sect. 4, we discuss related work and finally, we conclude in Sect. 5.

2 Maximum Common Subgraph Sampling

In this section, we describe how to extract maximum common subgraphs from a graph-based dataset. We start by introducing the necessary concepts in Sect. 2.1. In Sect. 2.2, we will present a high-level description of the polynomial algorithm

[1] National Cancer Institute: `http://cactus.nci.nih.gov/`

that computes a maximum common subgraph of two outerplanar graphs [4], which we will use for the feature generation method discussed in Sect. 2.3.

2.1 Graph Theoretical Concepts

We now formally define the necessary graph theoretical concepts. For an overview of graph theory, we refer to [6].

A labeled **graph** is a quadruple $G(V, E, \Sigma, \lambda)$, with V a finite set of vertices and $E \subseteq \{\{u, v\} \mid u, v \in V\}$ a set of edges. Σ is a finite set of labels and $\lambda : V \cup E \rightarrow \Sigma$ is a function assigning a label to each element of V ∪ E. If G is a graph, $V(G)$ denotes the set of vertices of G, $E(G)$ denotes the set of edges of G and λ_G denotes the labeling function of G. The **size** of a graph is a function mapping a graph to a real number $size(G) = \sum_{x \in V(G) \cup E(G)} w_{\lambda_G(x)}$, where each possible label of $l \in \Sigma$ has been assigned a weight w_l. In this article, we instantiate the size of a graph as the sum of its number of vertices and its number of edges, that is, we chose $w_{\lambda_G(x)} = 1$ for every $x \in V(G) \cup E(G)$.

A sequence $x_0, x_1, \ldots, x_n$ of vertices is a **path** from x_0 to x_n if and only if $\{x_i, x_{i+1}\} \in E(G)$, for all $i \in [0, n-1]$. A **cycle** $x_0, \ldots, x_n$ is a path such that $x_0 = x_n$. A path without repeated vertices is a **simple path**; a cycle without repeated vertices apart from the start and end vertex is a **simple cycle**. A graph G is **connected** if there is a path between any pair of its vertices; it is **biconnected** if for any two vertices u and v of G, there is a simple cycle containing u and v.

A graph is **planar** if it has a planar embedding, that is, it can be drawn in the plane in such a way that no two edges intersect except at a common vertex. The regions formed by the edges in a planar embedding are called **faces**. There is one unbounded face, which is called the outer face. A biconnected component or **block** of a graph G is a maximal subgraph of G that is biconnected. A **bridge** is an edge that does not belong to a block. An **outerplanar** graph is a planar graph that can be embedded in the plane in such a way that all of its vertices lie on the boundary of the outer face. An outerplanar graph consists entirely of blocks and bridges.

Figure 1(a) shows an example of a non-outerplanar graph in which there is one vertex (marked in light gray) that is not on the outside of the graph. The graphs in Fig. 1(b) and Fig. 1(c), however, are outerplanar. Every vertex is labeled with a color representing a chemical element: black for carbon, white for hydrogen and blue for nitrogen. Note that in all graphs, the edges between carbons and hydrogens are bridges, while the rest of the graphs form blocks. From a chemical viewpoint, blocks correspond to ring structures while bridges are linear fragments of the molecule.

Let G and H be graphs. G is a **subgraph** of H, if (i) $V(G) \subseteq V(H)$, (ii) $E(G) \subseteq E(H)$, and (iii) $\lambda_G(x) = \lambda_H(x)$ holds for every $x \in V(G) \cup E(G)$. Two graphs G and H are **isomorphic** if there exists a bijection $\varphi : V(G) \rightarrow V(H)$ such that for every $u, v \in V(G)$ the following holds: (i) $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$, (ii) $\lambda_G(u) = \lambda_H(\varphi(u))$, and (iii) if $\{u, v\} \in E(G)$ then $\lambda_G(\{u, v\}) = \lambda_H(\{\varphi(u), \varphi(v)\})$. A graph G is **subgraph isomorphic** to

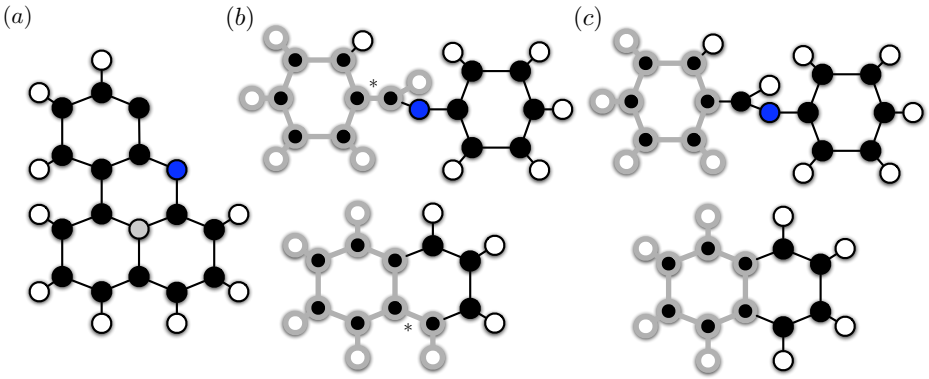

Fig. 1. Examples of molecular graphs. The colors of the vertices correspond to their labels: black for carbon, white for hydrogen and blue for nitrogen. (a) Example of a non-outerplanar graph, with the vertex that is not on the outside of the graph marked in gray. (b) A maximum common subgraph under the general subgraph isomorphism ($\text{MCS}_{\preceq}$), highlighed in gray. (c) A maximum common subgraph under the BBP subgraph isomorphism ($\text{MCS}_{\sqsubseteq}$), highlighted in gray.

H, denoted $G \preceq H$, if and only if G is isomorphic to a subgraph of H. The subgraph isomorphism problem, that is, the problem of deciding whether G is subgraph isomorphic to H, is NP-complete [7]; this also holds for outerplanar graphs.

A **block-and-bridge-preserving (BBP) subgraph isomorphism** from G to H is a subgraph isomorphism from G to H, denoted $G \sqsubseteq H$, such that (i) $\{u, v\} \in E(G)$ is a bridge iff $\{\varphi(u), \varphi(v)\} \in E(H)$ is a bridge, and (ii) $\{u, v\} \in E(G)$ belongs to a block iff $\{\varphi(u), \varphi(v)\} \in E(H)$ belongs to a block. That is, the BBP subgraph isomorphism is a special case of the general subgraph isomorphism in which the constraint holds that bridges of G are only mapped to bridges of H and edges of blocks of G only to edges of blocks of H. As opposed to the subgraph isomorphism problem, the BBP subgraph isomorphism problem is computable in polynomial time for outerplanar graphs [5]. For trees, which are special outerplanar graphs (they are block-free), the BBP subgraph isomorphism is equivalent to the subtree isomorphism.

A **common connected subgraph** I of two graphs G and H is a connected graph such that $I \preceq G$ and $I \preceq H$; it is a **maximum common connected subgraph** when in addition there exists no other common subgraph J, such that $size(I) < size(J)$. From now on we call this an $\text{MCS}_{\preceq}$ (where $\preceq$ means that it is mined under the general subgraph isomorphism) and implicitly assume that it is always connected. In the same way, we define an $\text{MCS}_{\sqsubseteq}$. Note that, since the BBP subgraph isomorphism is a more restricted version of the general subgraph isomorphism, an $\text{MCS}_{\sqsubseteq}$ will be subgraph isomorphic to one of the $\text{MCSs}_{\preceq}$. Note also that, in the worst case, there may exist a potentially exponential number of

MCSs. Interestingly, even though computing an $\text{MCS}_{\preceq}$ or an $\text{MCS}_{\sqsubseteq}$ between two general graphs is NP-hard [7], it is possible to compute an $\text{MCS}_{\sqsubseteq}$ between two outerplanar graphs in polynomial time by using the block-and-bridge-preserving (BBP) subgraph isomorphism [4].

Figure 1 shows a comparison between an $\text{MCS}_{\preceq}$ (b) and an $\text{MCS}_{\sqsubseteq}$ (c). In both examples, the MCS is highlighted in gray. Note that one of the edges is a bridge in the upper graph, while it belongs to a block in the lower graph (marked with a * in both graphs) and hence, it cannot be mapped under the BBP subgraph isomorphism. Chemically, it seems relevant not to map linear fragments to fragments that are part of a ring structure. This example shows that algorithms computing $\text{MCSs}_{\sqsubseteq}$ are more likely to generate smaller subgraphs than algorithms computing $\text{MCSs}_{\preceq}$.

For notational convenience, in the remainder of the text we will simply use MCS when we mean the $\text{MCS}_{\sqsubseteq}$.

2.2 Computing an MCS of two outerplanar graphs

In this section, we give a high-level description of the algorithm that computes an MCS of two outerplanar graphs [4]. The algorithm is based on a dynamic programming strategy that makes use of efficient matching algorithms during the partial solution building step. The two key procedures are: 1) *subgraph enumeration*, in which we will generate a set of particular subgraphs for each of the two input graphs and establish their parent-child relationships, and 2) *bottom-up MCS computation*. in which we will compute an MCS for each pair of generated subgraphs, building on the already computed solutions for pairs of children of these subgraphs.

Enumerating relevant subgraphs of an outerplanar graph First, we denote with G^r the rooted graph G where vertex $r \in G$, the root, is distinguished from the other vertices. Without loss of generality, we can assume that for all graphs, we have chosen a planar embedding. Given an outerplanar graph G^r, we introduce two kinds of subgraphs of G^r: the block-preserving-subgraphs and the block-splitting-subgraphs. The former are subgraphs in which a block is either entirely included in the subgraph or not; the latter are subgraphs which are created by removing part of a block between two vertices. We call these two kinds of subgraphs the **relevant subgraphs** of G^r. More formally,

- Given an outerplanar graph G^r, we denote with G^r_i the maximal connected subgraph of G^r containing r but none of the edges on the path(s) between r and a vertex i. Every subgraph G^r_i $(i, r \in V(G))$ is then a *block-preserving-subgraph* (BPS) of G. Remark that for several (r, i) pairs we may obtain identical BPSs. We note that $G^r_r = G^r$. Examples of BPSs can be found in Fig. 2(b) and Fig. 2(c).
- Given an outerplanar graph G^r, two vertices u and v in the same block of G and an orientation $o \in \{\curvearrowleft, \curvearrowright\}$ (counterclockwsie or clockwise), we denote with $G|_{o[u,v[}$ the maximal connected graph including u and v that remains

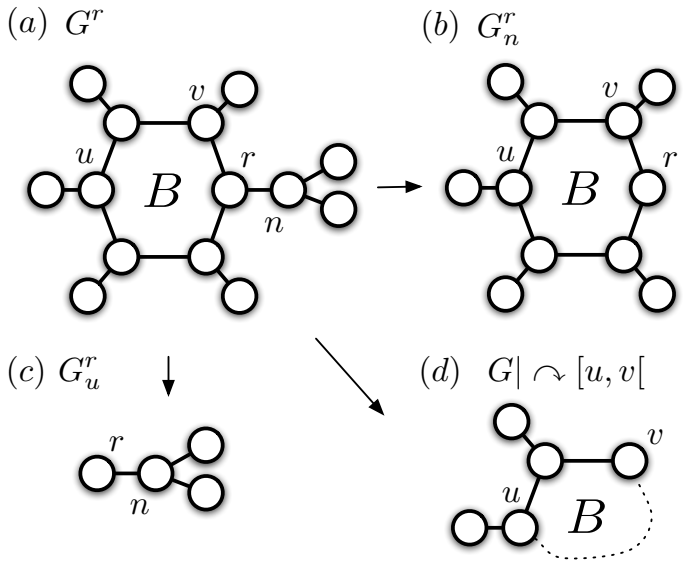

Fig. 2. (a) An (unlabeled) outerplanar graph G^r. (b) An example BPS G^r_n. (c) An example BPS G^r_u. (d) An example BSS $G|_{\curvearrowright[u,v[}$.

after removing the vertices between v and u on the Hamiltonian cycle[2] over all vertices of the block according to the orientation o, and removing all edges adjacent to v but not belonging to the block. We call these graphs *block-splitting-subgraphs* (BSS) of G^r. We use the notation"$[u, v[$" to stress that the edges of u and all vertices between u and v are kept while those of v are removed. Note that $G|_{o[u,u[} = G$. An example of a BSS can be found in Fig. 2(d).

We also define a *parent-child relationship* between the relevant subgraphs. We say that a relevant subgraph R_i is a child of a relevant subgraph R_j if (i) R_i is a strict subgraph of R_j, and (ii) there exists no other relevant subgraph R_k for which R_i is a strict subgraph of R_k and R_k is a strict subgraph of R_j.

We denote with $BPS(G)$ the set of all BPSs of an outerplanar graph G and with $BSS(G)$ the set of all BSSs of G. Finally, we define for two outerplanar graphs G and H the set $\mathcal{P}(G, H)$, containing the union of the set of pairs of their BPSs and the set of pairs of their BSSs:

$$\mathcal{P}(G, H) = (BPS(G) \times BPS(H)) \cup (BSS(G) \times BSS(H)).$$

As we want to make sure we process all $p \in \mathcal{P}(G, H)$ in increasing size, we will first order them lexicographically according to their size.

[2] A Hamiltonian cycle is a simple cycle which visits each vertex of the block exactly once.

Bottom-up MCS computation Once the relevant subgraphs have been determined, we compute an MCS using a bottom-up dynamic programming strategy.

When computing an MCS of two outerplanar graphs G and H, instead of considering all possible rooted graphs G_i^r $(r, i \in V(G))$ and H_j^s $(s, j \in V(H))$, it suffices to compute an MCS of G_i^r with i an arbitrarily chosen vertex from G with each graph in the set $\{H_j^s \mid s, j \in V(H)\}$ and to take the maximal one. Similarly, if suffices to choose one orientation for the BSSs of one graph and both orientations for the other one.

We now describe how to match two relevant subgraphs of the same type, that is, an BPS of G and a BPS of H or a BSS of G and a BSS of H. Matching two subgraphs of different types is not needed due to the BBP subgraph isomorphism. The key idea is to consider appropriate combinations of children, and to extend their MCSs into an MCS of their respective parents. The dynamic programming approach implies that we have access to the already computed MCS of all possible pairs of children.

In order to find an MCS of two BPSs G_i^r and H_j^s (whose roots have the same label), we make a weighted maximal matching between the set of children of G_i^r and the set of children of H_j^s (matching BSSs with BSSs and BPSs with BPSs, and checking that the connecting edges have identical labels) using Munkres' algorithm [8]. The weights represent the size of an MCS between the pairs of children.

In order to find an MCS of two BSSs $G|_{o^G[x_i,r[}$ and $H|_{o^H[y_j,s[}$ (splitting a block B_G and a block B_H respectively), where r and s have the same label, the MCS is the result of the best matching, that is, the matching that results in the largest MCS, between the MCS of the children $G|_{o^G[x_i,x_k[}$ and $H|_{o^H[y_j,y_l[}$ (in which the attached BPSs are also matched). If $\{r, x_i\}$ and $\{s, y_j\}$ are the only remaining edges of both blocks and they have identical labels, then $\{x_i, r\}$ is added to the MCS.

We note that, when computing the matchings in the above steps, only one solution is considered in case of ties. This results in a polynomial complexity, but if there are multiple possible MCSs, only one of them is returned.

One can prove that this algorithm runs in polynomial time by counting the number of children to consider and using known bounds on the running times of the algorithms used in the dynamic programming step e.g., a maximal matching can be computed in cubic time [8].

2.3 Method

In this section, we describe how we use the MCS algorithm to generate features for graphs. The idea is to select pairs of graphs from the dataset, and then compute one of their MCSs. Before we discuss the method, we give a problem description.

Problem description We define the task of generating features in graphs as follows. Consider a set of graphs $\mathcal{G}$, where each graph has been labeled positive

or negative w.r.t. a particular classification task:

$$\mathcal{G} = \{(g_i, C_i) \mid C_i \in \{+, -\}\}.$$

Given a dataset $\mathcal{G}$, a set of possible constraints c and a number k (with $0 < k < \infty$), the task is then to find a set of k subgraphs satisfying the constraints c that are used as features for $\mathcal{G}$.

MCS extraction First, we introduce some additional notation. We denote with $\mathcal{G}_+$ the subset of graphs belonging to the positive class, that is,

$$\mathcal{G}_+ = \{(g_i, C_i) \in \mathcal{G} \mid C_i = +\}.$$

In the same way, we define $\mathcal{G}_-$. Note that $\mathcal{G}_+$ and $\mathcal{G}_-$ form disjoint partitions of $\mathcal{G}$, that is, $\mathcal{G} = \mathcal{G}_- \cup \mathcal{G}_+$. Then, let $\mathcal{G}^*$ be the subset of outerplanar graphs of $\mathcal{G}$, that is,

$$\mathcal{G}^* = \{g \in \mathcal{G} \mid g \text{ is outerplanar}\}.$$

Now we are ready to introduce the notation for extracting MCS features. We define

$$\mathcal{F}(X, Y) = \{p \mid p = \text{MCS}(x, y),\ x \in X,\ y \in Y\}$$

where $\text{MCS}(x, y)$ returns *an* $\text{MCS}_{\sqsubseteq}$ of x and y computed and where X and Y are arbitrarily defined sets of graphs. Observe that per pair of graphs, we compute only one MCS. We can obtain different subsets of $\mathcal{F}(X, Y)$ by: 1) varying the selection strategy, that is, the way we choose X and Y, possibly using a sampling method, and 2) adding additional constraints on the found subgraphs p. In particular, the choices that we consider are:

1. **Selection strategies** on X and Y
   - $X = \mathcal{G}^*$ and $Y = \mathcal{G}^*$, that is, we compute all MCSs from all pairs of outerplanar graphs in our dataset;
   - $X = \mathcal{G}^*_+$ ($\mathcal{G}^*_-$) and $Y = \mathcal{G}^*_+$ ($\mathcal{G}^*_-$), that is, we consider only subgraphs common between graphs belonging to the same class (either positive or negative) in order to capture features that are more discriminative for the given target concept;
   - a sampling approach that selects couples (x, y) from $X \times Y$ uniformly at random. This allows one to trade-off the accuracy with the efficiency as a reduced set of k features can be generated more quickly. We denote a reduced set of k features as $\mathcal{F}^k$;
2. **Additional constraints c** on the retrieved subgraphs p
   - $freq(p, \mathcal{G}) \geq f$, that is, p is a subgraph of more than f graphs in the dataset $\mathcal{G}$;
   - $\chi^2(p, \mathcal{G}) > t$, that is p is a subgraph occurring in more graphs from the positive (negative) class than from the negative (positive) class, where the exact threshold is derived by the χ^2 score, often used to compute the significance of patterns [2];
   - $size(p) \leq s$, that is, p has to have a size below the threshold s.

When a constraint $\mathbf{c}$ is imposed on a set of features $\mathcal{F}$ we denote the resulting set as $\mathbf{c}(\mathcal{F})$. For instance, $freq(\mathcal{F}, \mathcal{G}) \geq f$ represents the set of patterns in $\mathcal{F}$ with a frequency higher than f. Furthermore, we use the notation $\arg\max_k \phi(\mathcal{F}, \mathcal{G})$ to denote the top-k features from $\mathcal{F}$, that is, the k features from $\mathcal{F}$ that score best with regard to a scoring function $\phi(\cdot)$.

Computational complexity In order to gain an understanding of the time complexity of the proposed approach, we identify and discuss four key processes: 1) the computation of the set of MCSs between two graphs x, y; 2) the selection strategy, which determines the set of graphs from which to sample the pairs (x, y); 3) the elimination of multiple occurrences of the same subgraph; and 4) the embedding of the extracted subgraphs in the graph dataset:

1. **MCS computation** While computing the MCS set under the general subgraph isomorphism is NP-hard, determining a single (random) MCS under the BBP subgraph isomorphism of two outerplanar graphs can be achieved in polynomial time using the algorithm discussed in Sect. 2.2.
2. **Selection strategy** While the extraction of the MCS set from all pairs of graphs in $\mathcal{G}^*$ invokes the MCS computation a number of times quadratic in the size of the set of examples, one can hope to achieve a good compromise by either a) randomly selecting a smaller subset of graphs in X, Y and invoking MCS(x, y) from all possible pairs, or b) directly selecting a smaller number of random graph pairs (x, y). This latter procedure raises an interesting question as to how the number of (distinct) subgraphs k and the number of graph pairs n relate (as the same MCS can be extracted from different graphs). This is investigated experimentally in Sect. 3, where we show that the relationship between k and n is just linear.
3. **Eliminating multiple MCS occurrences** To avoid multiple occurrences, we have to check for each new pattern whether it is isomorphic to an already found MCS. Because of the BBP subgraph isomorphism and the fact that all MCSs are outerplanar, this can also be realized in polynomial time and has to be repeated k^2 times with k the cardinality of the MCS set.
4. **Feature embedding** Once the set of k MCSs has been identified, the translation of these subgraphs into features is accomplished by doing a subgraph isomorphism test between each of the k elements in the MCS set and each of the m elements in $\mathcal{G}^*$. Using the BBP decomposition notion, this can be done in polynomial time for each of the $k \cdot m$ pairs. To compute the embeddings for the non-outerplanar graphs, that is, for every $g \in \mathcal{G} \setminus \mathcal{G}^*$, the (NP-complete) general subgraph isomorphism test can be used.

Hence, the overall complexity is polynomial in the size of the individual graphs, in the size of the graph set and in the size of the desired feature set (which is bounded by the square of the size of the graph set).

Notice that, in contrast to traditional local pattern mining approaches, the proposed technique does not require one to perform expensive embedding operations while searching for features, but only once the features have been generated,

that is, while local pattern mining techniques need to compute frequencies or correlation measures and therefore need to perform embedding computations *during* the search phase, our approach computes the embedding only *after* the whole set has been extracted.

In order to gain an understanding of the space complexity of the proposed approach, we identify and discuss two key processes: 1) the space requirements for the extraction of an MCS of two graphs and 2) the space requirements when processing the entire set of examples.

In the first case, the MCS algorithm requires to store a number of relevant subgraphs bounded by $O(m^2)$ with m being the number of vertices in the largest block. We note that in practice this does not imply a severe memory requirement for applications in chemoinformatics.

In the second case, the key process is the check for multiple MCSs occurrences. For this we need to keep track of all unique MCS patterns found. In Sect. 3, we empirically show that the number of unique MCS features grows linearly w.r.t the number of examples and not quadratically as one would intuitively expect. This property allows us to conclude that the memory requirements are in practice linear w.r.t. the dataset size.

A C++ implementation of the presented method can be downloaded at `http://www.cs.kuleuven.be/~dtai/PMCSFG`.

3 Experimental Evaluation

In this section, we perform an experimental analysis to measure the quality of the patterns under the various parametric choices and the computational time needed to generate them. The properties and the quality of the extracted subgraphs are evaluated by using them as features in predictive tasks for 60 problems from chemoinformatics. We compare the results against several related state-of-the-art methods and provide a discussion.

3.1 Datasets

The NCI dataset collection has been made publicly available by the National Cancer Institute and provides screening results for the ability of thousands of compounds to suppress or inhibit the growth of a panel of 60 human tumor cell lines. The datasets used here correspond to the parameter GI_{50}, the concentration that causes 50% growth inhibition. For each cell line, approximately 3,500 compounds are provided together with information on their cancer-inhibiting action, which defines a binary classification problem. We use the datasets of Swamidass et al. [9], which are available from the authors upon request.

Each molecule is described in the Tripos Sybyl MOL2 format.[3] From this we extract a graph in which each vertex corresponds to an atom and each edge to a bond. The vertices are labeled with general atom types (e.g., N, C) and the

[3] `http://www.tripos.com/data/support/mol2.pdf`

edges are labeled single, double, triple, amide or aromatic. Hydrogen atoms are dropped.

3.2 State-of-the-art methods

We compare our method against three state-of-the-art methods that construct features for graphs: one method that performs correlated subgraph mining [2] and two methods that exhaustively enumerate all possible subgraphs [3, 10]. We describe these methods using the notation introduced in Sect. 2.3.

First, we consider a correlated graph miner [2], which traverses a search space in order to find the top-k correlated graph patterns. Here, each pattern receives a χ^2-correlation score w.r.t. the class value. It is known that correlated subgraph miners outperform frequent subgraph miners, which mine patterns under the frequency constraint, in terms of predictive performance [2]. We call this method C-GP. In our notation, it corresponds to $\arg\max_k \chi^2(\mathcal{A}, \mathcal{G})$, where $\mathcal{A}$ denotes the set of all possible graphs.

Second, we consider the method proposed by Wale et al. [3], which generates all possible graph patterns that occur at least once in the dataset. The subgraph size is upper-bounded by a user defined parameter s. Wale et al. [3] have shown that their method outperforms earlier methods such as graph kernels and fingerprints. We call this method A-GP. In our notation, it corresponds to $size(freq(\mathcal{A}, \mathcal{G}) \geq 1) \leq s$.

Third, we consider a method that computes the FP2 fingerprints (generated using OpenBabel v2.1.1[4]). This is an exhaustive method that generates all possible paths (linear sequences) up to length $s = 7$. Moreover, it makes use of basic chemical knowledge to label paths linked to a cycle and to discard uninformative paths. Because even for small s (say 7 or 8) this rapidly leads to vast numbers of features, the generated features are typically compressed into a fingerprint using a kind of hashing of the the occurrences of the paths onto a fixed-length vector [10]. In this step, information is lost as it becomes impossible to find out which patterns are involved in the fingerprint. Despite this drawback, fingerprints are considered state-of-the-art among chemists [10]. We call this method FP2. In our notation, it corresponds to $size(freq(\mathcal{P}, \mathcal{G}) \geq 1) \leq s$, where $\mathcal{P}$ denotes the set of paths.

3.3 Methodology and parameter settings

We consider a variety of parametric choices as detailed in Sect. 2.3. Table 1 gives an overview of the variants that will be investigated as well as an overview of the state-of-the-art methods we will compare to.

A-MCS corresponds to extracting MCSs from all outerplanar examples, while P-MCS and N-MCS only extract MCSs from positive or negative examples alone, respectively. R-MCS corresponds to extracting MCSs from randomly sampled pairs of outerplanar graphs, while F-MCS and C-MCS first extract all MCSs and then keep the top-k ones w.r.t. frequency and χ^2, respectively.

[4] `http://openbabel.sourceforge.net`

Table 1. Overview of the different parametric choices for $\mathcal{F}$ and the state-of-the-art methods.

Abbr.	Method	Language	Hashing
MCS variants			
A-MCS	$\mathcal{F}(\mathcal{G}^*, \mathcal{G}^*)$	graphs	no
P-MCS	$\mathcal{F}(\mathcal{G}^*_+, \mathcal{G}^*_+)$	graphs	no
N-MCS	$\mathcal{F}(\mathcal{G}^*_-, \mathcal{G}^*_-)$	graphs	no
R-MCS	$\mathcal{F}^k(\mathcal{G}^*, \mathcal{G}^*)$	graphs	no
F-MCS	$\arg\max_k freq(\mathcal{F}(\mathcal{G}^*, \mathcal{G}^*), \mathcal{G})$	graphs	no
C-MCS	$\arg\max_k \chi^2(\mathcal{F}(\mathcal{G}^*, \mathcal{G}^*), \mathcal{G})$	graphs	no
State-of-the-art methods			
C-GP	$\arg\max_k \chi^2(\mathcal{A}, \mathcal{G})$ [2]	graphs	no
A-GP	$size(freq(\mathcal{A}, \mathcal{G}) \geq 1) \leq s$ [3]	graphs	no
FP2	$size(freq(\mathcal{P}, \mathcal{G}) \geq 1) \leq s$ [10]	sequences	yes

Parameter settings for MCS variants For R-MCS, F-MCS and C-MCS, we chose $k = 1000$. Since R-MCS is a non-deterministic method, it was always run 10 times and boxplots are reported. For F-MCS and C-MCS, we also chose $k = 1000$. For all MCS methods, we discard subgraphs that only have a single vertex, as was done by [2].

Parameter settings for state-of-the-art methods For C-GP, we chose $k = 1000$ and mined the top-1000 most correlated patterns in the training data. For A-GP, we consider all subgraphs from length 1 to 7, that is, we chose $s \leq 7$, as recommended by [3]. FP2 also uses $s \leq 7$ and requires one additionally to specify the number of bits for the pattern encoding vector. A common choice for this number is 10, and since 1024 (the length of the vector) is closest to the value of k, we adopt the same value of 10.

Evaluation Since we want to investigate the predictive quality of different feature generation methods, we vary only the feature generation step and resort to the same classification procedure for all methods.

Given a graph dataset $\mathcal{G}$, we first generate features only from the training set. Then, we propositionalize each example in $\mathcal{G}$ to a one-bit vector encoding representation: given a feature set of size k, each graph $g \in \mathcal{G}$ is encoded as a k-dimensional binary vector, where a 1 is marked in the i-th position if the i-th subgraph is subgraph isomorphic to g. The general subgraph isomorphism is used for this matching for all methods.

As classification model we use SVMs in combination with the Tanimoto-kernel [9]:

$$\mathcal{K}_T(x, y) = \frac{\sum_{i=1}^{N}(x_i = 1 \wedge y_i = 1)}{\sum_{i=1}^{N}(x_i = 1 \vee y_i = 1) - \sum_{i=1}^{N}(x_i = 1 \wedge y_i = 1)}$$

In words, this kernel computes a similarity between vector x and vector y by counting the number of common patterns (i.e. the set-intersection) between the two molecules as a fraction of the total number of patterns that occur in both molecules (i.e. the set-union). The Tanimoto-kernel is considered state-of-the-art for the classification of small molecules [10]. As implementation we used SVM^{light} [11].

To evaluate the classification models, we use the area under the ROC curve (AUROC) score [12] and the H score introduced by Hand [13], who shows that AUROC fails to take into account the relative costs of misclassifications of different classifiers. The H score does not suffer from this problem.

For all experiments, a stratified 10-fold cross-validation is used. The regularization parameter of the SVM is tuned out of 10 values running an internal 5-fold cross-validation over the training data.

We compute the statistical significance of the different methods in two ways. On the one hand, generalization over datasets follows from the win/loss-ratio. In particular, we use the Friedman test combined with a Nemenyi post-hoc test to compute significance [14]. The Friedman test is a non-parametric test for statistical comparisons of classifiers over multiple datasets. It ranks the algorithms for each dataset separately, with the best performing algorithm getting the rank of 1. In case of a tie, the average rank of the tied models is assigned. Then, a Nemenyi post-hoc test is used to analyze which of the classifier's ranks differ significantly from each other: the performance is significantly different if the corresponding average ranks differ by at most the critical difference, which depends on the significance level and the number of classifiers [14].

We also use a second statistical test, which shows how well different classifiers are able to generalize to other instances from the same population. One classifier is significantly better than another at the 1% level for samples of ≈3,500 molecules when an increase of ≈2.5% in AUROC or H is measured.

3.4 Results

We organize the experimental results as answers to a set of six questions:

Q1 What is the predictive quality of MCS features obtained under different selection strategies?

Q2 What are the effects of applying different constraints on the obtained MCS features?

Q3 How does the quality of the feature set vary w.r.t. the number of sampled MCS features?

Q4 How many pairs of molecules need to be sampled in order to obtain k unique MCS features?

Q5 How does MCS feature construction compare with state-of-the-art feature construction methods?

Q6 What are the runtimes of the MCS feature generation methods and how do they compare with state-of-the-art feature construction methods?

Table 2. Average scores and ranks for AUROC and H over 60 datasets when comparing different selection strategies.

	AUROC		H	
Method	Average	Average rank	Average	Average rank
A-MCS	0.796	1	0.301	1.02
P-MCS	0.792	2.08	0.294	2.05
N-MCS	0.788	2.92	0.286	2.93
Critical difference for the average ranks at the 1% significance level: 0.53				

Q1: What is the predictive quality of MCS features obtained under different selection strategies? Figure 3 shows the predictive performance (AUROC and H) for A-MCS, P-MCS and N-MCS. On average, A-MCS resulted in approximately 7800 patterns, P-MCS in 4500 patterns and N-MCS in 3200 patterns. According to the Friedman test, for which the average ranks and critical difference are shown in Table 2, A-MCS is significantly outperforming P-MCS for both evaluation measures. P-MCS in turn outperforms N-MCS for both measures.

However, the average AUROC and H scores over the 60 datasets (also shown in Table 2), differ for less than 2.5%. This is an interesting result for the practitioner since, when there are reasons to believe that the negative class is more complex to model or when the available dataset exhibits a larger number of examples from the negative class (conditions that often occur in chemoinformatic activity prediction tasks), one can resort to sampling from positive examples alone without losing much predictive performance in practice.

In conclusion, while according to the Friedman test, extracting MCS features of positive (or even negative) examples results in significantly worse predictive performances, the second statistical test, which generalizes over molecules from the same population, indicates that there is no significant difference between the methods. Further investigation suggests that the small decrease in performance of P-MCS and N-MCS is caused by the reduced number of features (see also Q3).

Q2: What are the effects of applying different constraints on the obtained MCS features? To answer this question, we will compare a random sample of 1000 MCS features (R-MCS), that is, applying no constraint at all, to the 1000 most frequent MCS features (F-MCS) and the 1000 most correlated MCS features (R-MCS). The predictive performance of R-MCS, F-MCS and C-MCS is shown in Fig. 4. Note that, because for R-MCS the results are averaged over 10 runs, boxplots are shown. These show that, despite the non-deterministic nature of the procedure, R-MCS achieves quite stable results.

The Friedman test (results shown in Table 3) shows a clear advantage for R-MCS over F-MCS and C-MCS. However, again the average AUROC and H scores indicate no significant difference (Table 3).

Table 3. Average scores and ranks for AUROC and H over 60 datasets when applying different constraints.

	AUROC		H	
Method	Average	Average rank	Average	Average rank
R-MCS	0.784	1	0.280	1.02
F-MCS	0.774	2	0.263	1.98
C-MCS	0.761	3	0.244	3
Critical difference for the average ranks at the 1% significance level: 0.53				

In conclusion, the results show that extracting MCS features from randomly sampled pairs of examples does not perform significantly worse than applying a frequency or correlation constraint on the MCS features. This is a surprising result, since randomly sampling 1000 MCSs is less computationally expensive (see Q4) than mining *all* MCSs and then post-processing those under some constraint to obtain the 1000 best features. A possible reason for this is that constraints tend to decrease the diversity of the set, i.e. features that are highly frequent or correlated with the target could be highly inter-correlated and hence redundant and ultimately uninformative. We further investigate this issue in the following section.

Redundancy issues In order to gain a deeper understanding on the quality and the differences between the various feature sets, we define some indicators reported in Table 4. First, we define *uniqueness* as the percentage of examples with a bit-vector encoding that is unique, i.e. different from that of all the other examples in the dataset. It is evident that examples having the same encoding cannot be further discriminated by any classification method. Hence, a high uniqueness is a desirable property.

Second, we report the generalization of the mutual information measure, i.e. the *total correlation* [15] (also known as the multivariate constraint or multi-information) to express the amount of redundancy existing among the set of features considered as random variables. The total correlation (TC) is defined as: $TC(X_1, X_2, \ldots, X_k) = \sum_i^k H(X_i) - H(X_1, X2, \ldots, X_k)$ for the set of k features, where $H(\cdot)$ is the (joint) information entropy. It represents the amount of information shared among the variables in the set. The sum $\sum_i^n H(X_i)$ represents the amount of information (in bits) that the features would possess if they were totally independent of one another. The term $H(X_1, X_2, \ldots, X_n)$ is the actual amount of information that the feature set contains. The difference between these terms therefore represents the absolute redundancy present in the given set of features, that is, the TC tells us how related a group of features are. A near-zero TC indicates that the features are essentially statistically independent; they are completely unrelated, in the sense that knowing the value of one feature does not provide any clue as to the values of the other features. A maximum value for TC is achieved when one of the features is completely redundant with respect to all of the other features.

Table 4. Redundancy evaluation of the different feature construction methods (averaged over 60 datasets) that yield 1000 features.

Method	Uniqueness	Total Correlation
A-MCS	99.19± 0.18	N/D
R-MCS	98.52± 0.26	103.55 ± 1.26
F-MCS	97.00± 0.45	148.57 ± 2.74
C-MCS	91.38± 2.11	139.32 ± 2.59
A-GP	99.36± 0.20	N/D
C-GP	53.92± 5.74	212.44 ± 16.65

Table 5. Distribution of the number of edges of three feature generation methods for a representative dataset (786_0). O.k represents the $n \cdot k$ order statistic of the distribution.

Method	Average	Minimum	O.05	O.25	O.5	O.75	O.95	Maximum
A-MCS	13.22 ± 7.56	1	5	9	12	16	27	98
A-GP	6.46 ± 0.86	1	5	6	7	7	7	7
C-GP	9.86 ± 3.80	1	5	8	10	12	16	19

We argue that a good set of features should have 1) a high uniqueness (so to be injective and not commit to some predefined, target independent equivalence notion between examples) and 2) a low total correlation, i.e. a low amount of information shared among the features.

All results have been averaged over the 60 datasets and only test set examples, that were not used for the generation of the patterns, were considered. Since we do not have access to the actual patterns that were generated by FP2, this method is not included in the table. Because the total correlation of different features sets only has a valid interpretation when dealing with the same amount of features, we do not report it for the 10^5 features of A-GP or for the 7800 features of A-MCS. According to these indicators, R-MCS selects a better set of features than the other strategies. Moreover, we have also observed that R-MCS returns features with a high frequency (occurring on average in 1/3 of the test set), showing that with high probability, computing MCSs between randomly chosen pairs of graphs leads to features that are also frequent.

We finally report in Table 5 some order statistics on the edge set size distribution of the subgraphs retrieved with A-MCS, A-GP and R-GP. A-MCS shows a clear preference in selecting significantly larger (and perhaps more interesting) subgraphs.

In conclusion, the features generated by A-MCS and R-MCS seem to have a larger uniqueness and are less redundant, which likely contributes to their superior predictive performance.

Q3: How does the quality of the feature set vary w.r.t. the number of sampled MCS features? We measure the quality of the feature set as the

Table 6. Predictive performance (AUROC) on 5 NCI datasets with an increasing number of randomly sampled MCSs.

	Number of MCS features						
Dataset	100	200	400	800	1600	3200	6400
SNB_19	74.3	76.0	77.2	78.0	78.8	79.2	79.4
M14	74.3	76.0	77.6	78.7	79.5	80.0	80.2
NCI_H522	74.9	76.3	77.6	78.7	79.5	80.1	80.3
786_0	75.1	77.1	78.3	79.4	80.1	80.4	80.7
HCT_116	76.2	78.0	79.4	80.4	81.2	81.7	82.0

predictive performance over 5 randomly selected datasets as we increase k, the number of randomly sampled MCSs, from 100 to 6400 (each result has been averaged over 10 runs). For this experiment we do not tune the regularization parameter of the SVM, but take a fixed value equal to 1 (this was the best-performing parameter value in the previous experiments). Table 6 shows an AUROC improvement of ≈5% when increasing the number of patterns from 100 to 6400 with a saturation level around 3200 patterns.

In conclusion, our intuition that using more features boosts predictive performance is correct. Note that with very few patterns, it is already possible to obtain a reasonable predictive performance.

Q4: How many pairs of molecules need to be sampled in order to obtain *k* unique MCSs? We experimentally determine the functional link between the number of examples and the number of unique MCSs by considering two strategies. In the first strategy, we take subsets of n examples and consider all $n(n-1)/2$ possible pairs of which we compute an MCS (S1). In the second strategy, we consider a random sample of m pairs from the set of all examples (S2). This corresponds to R-MCS.

The results of the two strategies are reported in Fig. 5. We observe that S2 needs to consider less pairs to obtain the same amount of unique MCSs, confirming the intuition that the repeated use of the same molecule in different pairs yields a smaller number of unique MCSs. Specifically, we found that in order to obtain 1000 different MCSs we need 45,000 pairs of randomly sampled molecules or a random sample of 400 molecules out of which to consider all possible pairs. We have observed an almost perfect linear relationship (with coefficient 2.6) between the number of molecules and the number of different MCSs, that is, given a set of 1000 molecules, extracting the MCSs from all pairs gives 2,600 unique MCSs.

The reason is that the number of distinct MCSs does not grow linearly, but rather as the square root of the number of pairs of examples as shown in Fig. 5. The explanation for this behavior is subject of current study, but it seems to be

Table 7. Average scores and ranks for AUROC and H over the 60 NCI datasets for the state-of-the-art feature generation methods.

	AUROC		H	
Method	Average	Average rank	Average	Average rank
A-MCS	0.796	1.45	0.301	1.35
A-GP	0.796	1.55	0.299	1.65
R-MCS	0.784	3.18	0.280	3.15
FP2	0.779	3.82	0.270	3.85
C-GP	0.684	5	0.134	5
Critical difference for the average ranks at the 1% significance level: 0.94				

related with the specific highly combinatorial nature of subgraphs[5] which biases shorter subgraphs to occur exponentially more frequently, which in practice, greatly reduces the number of different MCSs actually present.

In conclusion, when considering a sampling approach, it is better to take the full set of examples into account and consider random pairs, rather than computing MCSs of all pairs on a selected subset of examples.

Q5: How does MCS sampling compare with state-of-the-art feature generation methods? We first compare R-MCS (results are again averaged over 10 runs) to C-GP over the 60 datasets. Figure 6 shows a clear advantage for R-MCS. Also the Friedman test and the average AUROC and H scores (Table 7) show that R-MCS is performing significantly better than C-GP.[6]

Next, we compare R-MCS with FP2 (Fig. 6). Here, the Friedman test as well as the average AUROC and H scores show that there is no significant difference between these two methods (Table 7).

Finally, we compare A-MCS with A-GP. Figure 6 shows that both methods are competitive in terms of predictive performance. The outcome of the Friedman test and the average AUROC and H scores (Table 7) also show that the performances of A-MCS and A-GP are not significantly different. However, A-GP needs ≈150,000 patterns to reach this performance, while A-MCS needs only ≈7,800 patterns. Moreover, it can be argued that, because of the BBP subgraph isomorphism, the patterns of A-MCS are more easily interpretable from a chemical viewpoint.

To further investigate the quality of both sets of patterns, we have randomly selected 1000 patterns from the approach of Wale et al. [3] (R-GP) and compared the decrease in predictive performance. For R-GP, the average AUROC was

[5] A similar behavior is observed in the growth of the number of distinct words (which are sequences of atomic letters) in natural texts.

[6] Bringmann et al. [2] argue that mining the top-k sequences introduces less redundancy in the patterns than mining the top-k graphs. We have therefore tested the latter approach which yields an average AUROC of 73.6, still significantly below that of R-MCS.

0.679, while the average H score was 0.131. It turns out that GP degrades much more than MCS: the decrease in average AUROC and H (A-GP – R-GP) is 11.7 and 16.8, while for our approach (A-MCS – R-MCS) it is only 1.1 and 2.1, respectively. This shows that MCS features are more robust and meaningful. To check the redundancy of the patterns generated by R-GP, we also computed their uniqueness (25.25± 2.24) and redundancy (12.28 ± 0.80) . These numbers show that R-GP yields a set of nearly totally independent features (see Table 4). However, if we compare the predictive performance of R-MCS to the one of R-GP, it also becomes clear that achieving non-redundancy among the features is not the only prerequisite to generate a good set of features.

In conclusion, A-MCS and R-MCS can be considered as state-of-the-art feature generation methods.

Q6: What are the runtimes of the MCS feature generation methods? We executed A-MCS and R-MCS on an Intel Core2Quad Q9550 CPU (2.8GHz) for a representative set of 3910 NCI molecules with an average 23 vertices and 25 edges. First, we compare A-MCS with R-MCS. A-MCS needed $2.8 \cdot 10^5$ seconds, while R-MCS needed 2,142 seconds. Obviously, A-MCS is a time-consuming task, partly because of the many tests for duplicate MCSs. R-MCS, however, has a good trade-off between predictive performance and efficiency: it is 175 times faster, while only a decrease of 1.1% in AUROC was measured. One argument in favor of A-MCS, however, is that it, unlike the other feature generation methods, can easily be run in parallel.

Second, we compare R-MCS with C-GP. We randomly selected 5 datasets from the 60 NCI datasets for this experiment. R-MCS needed on average 2,327 seconds per dataset, while C-GP needed on average 54,322 seconds, which is 23 times slower than R-MCS.

In conclusion, when handling large datasets or runtimes are important, R-MCS provides a good trade-off between predictive performance and efficiency. Moreover, R-MCS achieves a speed-up of a factor 23 w.r.t. a typical correlated graph miner.

3.5 Discussion

We have shown that features obtained as the maximum common subgraphs from all pairs of instances in a dataset (or those obtained by sampling from a reduced set of pairs) allow the construction of predictive models achieving state-of-the-art performance on several tasks from chemoinformatics. There are however some drawbacks and implications of the presented method that deserve to be further discussed.

First, we notice that efficiency in the proposed approach can be guaranteed only when restricting to outerplanar graphs. Indeed, if an instance is a non-outerplanar graph, then it is not considered in the feature generation process. This restriction is not particulary severe when (a) the proportion of non-outerplanar examples is very small (few percentages w.r.t. the entire dataset

size) or (b) the number of cases where interesting features are themselves non-outerplanar is negligible. The first case is often true in many chemoinformatic applications although there exist datasets where the number of non-outerplanar instances is relatively large (10-20% of the total size. In those cases, methods that can exploit all the available material could in principle achieve better performance by simple virtue of a larger data set from which to extract relevant features. We have experimentally investigated the consequences of point (b) by extracting the top-k correlated subgraphs according to the graph miner of [2]. In this case, we have verified that all subgraphs are indeed outerplanar. A possible explanation for this is that the found patterns are too small to form non-outerplanar examples.

Second, we observe that by using the BBP subgraph isomorphism, ring structures will be either entirely selected as part of the MCS or not at all. As a consequence, ring structures and linear fragments are treated in a different way. This bias seems to have positive effects on the quality of the retreived patterns when dealing with applications from chemoinformatics as was experimentally shown in [4]. The effect of this bias on graphs in other types of domains needs to be empirically evaluated on a per-application basis.

Third, we acknowledge that extracting MCS features from all possible pairs of instances is a quadratic procedure which therefore does not scale well when dealing with large datasets. To tackle this issue, we have proposed a randomization strategy that sacrifices predictive performance in order to speed up the process. Interestingly, we have experimentally shown that the performance of models build on the MCS features saturates rapidly with the number of different MCSs so that only a relatively small number of random pairs of instances is needed to achieve results comparable with the all-pairs case. Once again though, it is unclear if these findings would hold true in different domains.

4 Related Work

This work is related to various streams of research. Firstly, our technique can be regarded as propositionalizing a relational or graph-based representation as is common in logical and relational learning [16]. Various techniques have been used to generate features of interest [17]. Our approach differs from these propositionalization approaches in that it works bottom-up and also that it computes pairwise minimally general generalizations of the examples that can be used as features, and it combines this idea with randomization. It is straightforward to adapt our technique for use in logical and relational learning. One only has to replace the use of the maximum common subgraph notion by a relational notion of minimally general generalization. Two such frameworks are well-known [16]: when working under θ-subsumption, the minimally general generalization is unique and is therefore called the least general generalization [18], while when working under OI-subsumption [16] it is – as the MCS – not necessarily unique. On the other hand, the size of the least general generalization under θ-subsumption of two objects may be as large as the product of the sizes

of these objects, while the size of the minimally general generalization under OI-subsumption is bounded by the size of the objects themselves. The differences between OI-subsumption and θ-subsumption are akin to those between subgraph isomorphism and homomorphism. While the use of subgraph isomorphism is more common when working with graphs, in inductive logic programming, the alternative notion, based on homomorphism is more common.

There are different ways to define and compute maximum common subgraphs. A large number of approaches can be related to distance metrics [19]. Raymond et al. [20] give an elaborate overview of existing similarity measures for molecules that are graph-based. Most of these algorithms avoid the computational complexity by computing approximate values, as the maximum common subgraph problem is in general NP-hard. In our work, we use an alternative matching operator, that is, the block-and-bridge-preserving subgraph isomorphism, which runs in polynomial time and is suitable for molecules [4]. An alternative approach to reduce the complexity could be to consider common substructures which can be computed more easily, such as multisets of common vertex labels [21]. However, an important drawback is that the more complex shared substructures are not taken into account.

Secondly, our work is related to the common practice in constraint-based graph mining, where constraints on subgraphs of interest are formulated and all subgraphs satisfying the constraints are generated. A wide variety of different constraints has been considered in graph mining, such as frequency-based [1, 22], generality-based, using one or two classes, imposing syntactic constraints, and combinations of those with particular subclasses of patterns such as paths [23]. In addition, there has been research on correlated pattern mining [2], where the goal is to find the top-k patterns according to a statistical significance measure such as χ^2 or information gain. In both types of approaches, one typically performs a complete search. This leads one to finding all solutions satisfying the constraints. While completeness and optimality are interesting theoretical properties, these approaches are also computationally much more demanding and may be harder to tune (that is, set parameters) than the simple randomized approach we pursued. At the same time, the completeness and optimality properties are not directly related to the true task in these graph miners, which is concerned with finding good representations of the graphs or molecules for use in classification. Our work shows clearly that – at least for molecular applications – a much simpler approach without strong guarantees may well achieve better results both in terms of predictive performance and efficiency.

It is interesting to note that in [24] the authors propose to rank the subgraphs returned by a frequent graph miner according to a notion of statistical significance[7] and show that in a chemical database the selected features are typically subgraphs that are in fact the "largest common subgraphs in a class of medically effective compounds".

[7] The p-value for a subgraph there is defined as the probability that the given subgraph occurs in a database of random graphs with a support higher than the observed frequency.

The favorable properties of randomization approaches, in particular the fact that choosing random features can be better than choosing them according to specific criteria, have already been noted in other contexts e.g., for selecting features in distance construction [25]. Recently, the randomization idea has also been suggested in the area of pattern mining. Chaoji et al. [26] have introduced a feature construction method that obtains good patterns by sampling under diversity constraints. However, the suggested method requires the user to tune and specify two parameters that control the diversity (orthogonality) and representativeness respectively.

We conjecture that methods looking for patterns that satisfy given constraints are more subject to redundancy issues than randomized methods. The intuition here is that similar or correlated patterns do exhibit the same properties w.r.t. the constraints and are therefore more likely to be all selected in the top-k set, hereby reducing the diversity of the set. Intuitively, the randomization procedure decreases the chance to select two patterns that are related in any special way (e.g., being similar or correlated). At the same time, a randomization procedure should not decrease the quality of the retrieved patterns. In the top-frequency case, sampling k elements randomly from a larger set of top-frequent patterns leads to patterns with a lower frequency on average than those obtained by a direct top-k frequent approach. Hence, the random sampling has a negative impact on the desired pattern quality, that is, the selected patterns are less frequent and potentially less relevant. In the MCS case, the random sampling does not alter the main property of a pattern of being the maximum common subgraph between a pair of instances.

Thirdly, as shown by De Raedt and Ramon [19], the notion of a minimally general generalization is closely related to that of a distance measure. For instance, the notion of maximum common subgraph under subgraph isomorphism is used in Bunke and Shearer's distance measure [27], while the one we are using (based on BBP-subgraph isomorphism) was used by Schietgat et al. to construct a metric [4]. Furthermore, as kernels can be viewed as a kind of similarity measure our work is also related to kernels. Many types of graph kernels [28–30] correspond to some feature space where every possible subgraph corresponds to a feature. The kernel then counts how many subgraphs two examples have in common. The subgraphs are typically defined in such a way that they can be enumerated in an implicit way such that the counting procedure can be done efficiently (e.g. via dynamic programming procedures). In these cases however, as the dimension of the feature space associated with the kernel becomes exponentially larger, there is an increasing probability that a significant fraction of the feature space dimensions will be poorly correlated with the target function. As a consequence, even when using large margin classifiers, one can fail to obtain models with good generalization performance [31]. In order to tackle these issues, several remedies have been proposed, from down-weighting the contribution of larger fragments and/or bounding a priori their size, to a direct manipulation of the Gram matrix. Alternatively one can try to identify a strong bias, relevant to the task at hand, and consider only a selected subset of structures to limit the

dimension of the feature space without degrading the prediction performance. Our approach follows the latter strategy in that it generates at most a single feature per pair of examples and a relatively small set of randomly chosen pairs is sufficient in practice to achieve good performance. We empirically showed that the bias of the MCS operator seems to be very well suited for chemoinformatic tasks.

A final stream of research is related to chemoinformatics, where the most common state-of-the-art approach to feature construction in molecules is to generate all patterns of size up to k (typically paths) that occur in at least one molecule [3, 10], the so-called fingerprints. The differences with our approach are that our features are guaranteed to occur in at least two molecules, that they are typically also much more informative as their size is typically larger, and at the same time, the number of such features is much smaller.

5 Conclusions and Future Work

We have introduced a simple, direct and effective approach to extracting patterns in graphs. It is based on the idea of computing the maximum common subgraph of randomly selected pairs of graphs. The approach is very efficient (it runs in polynomial time thanks to the restriction to outerplanar graphs), it does not require specifying any extra parameter (since one can simply extract all possible distinct pairwise MCSs), yields better sets of features than alternative approaches (as measured by the predictive performance of classifiers built using the returned subgraphs as features) and seems to produce a smaller and less redundant set of features than alternative techniques.

It was argued that the minimally general generalization approach provides an interesting alternative to the fingerprints that are so popular in chemoinformatics today. The advantages being that one obtains significantly larger and hence, chemically more meaningful patterns, as well as a smaller number of them. We intend to further investigate this idea in a chemoinformatics context.

Probably the most surprising finding of our work was that extracting MCSs randomly produces better features than more traditional and computationally more demanding all solutions or top-k approaches in graph mining. This in turn sheds new light on the traditional local pattern mining approach, which has dominated the field of data mining in the past 15 years. Our results indicate that for some tasks, such as finding interesting and representative features in molecular data it may be better to employ simpler and more efficient approaches based on e.g., randomization. Therefore, we hope that this work encourages more research in this direction.

Future work includes 1) the analysis of maximum common subgraph under BBP-isomorphism for direct use as a graph-kernel; 2) the extension of the maximum common subgraph notion to non-connected components; 3) the exploration of different sampling strategies to further reduce computational costs; and 4) the use of different language bias e.g. maximum common subtrees or subsequences, for increased efficiency.

Acknowledgments

L.S. is supported by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT). F.C. is supported by the GOA Probabilistic Logic Learning. J.R. is partially supported by the Fund for Scientific Research (FWO) of Flanders and the ERC Starting Grant 240186. The authors thank Björn Bringmann and Albrecht Zimmermann for their software and for valuable suggestions. The authors thank Nikil Wale as well for providing his software and Hendrik Blockeel, Maurice Bruynooghe and Kurt De Grave for fruitful discussions. This research was conducted using high-performance computational resources provided by the K.U.Leuven (`http://ludit.kuleuven.be/hpc`).

References

1. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), Japan, IEEE Computer Society (2002) 721–724
2. Bringmann, B., Zimmermann, A., Raedt, L.D., Nijssen, S.: Don't be afraid of simpler patterns. In: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases. (2006) 55–66
3. Wale, N., Watson, I., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems **14** (2008) 347–375
4. Schietgat, L., Ramon, J., Bruynooghe, M., Blockeel, H.: An efficiently computable graph-based metric for the classification of small molecules. In: Proceedings of the 11th International Conference on Discovery Science. Volume 5255 of Lecture Notes in Artificial Intelligence. (2008) 197–209
5. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. In: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA (August 2006) 197–206
6. Diestel, R.: Graph Theory. Springer-Verlag (2000)
7. Garey, M.R., Johnson, D.: Computers and Intractability: a Guide to the theory of NP-Completeness. Freeman and Co. (1979)
8. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5**(1) (1957) 32–38
9. Swamidass, S.J., Chen, J., Bruand, J., Phung, P., Ralaivola, L., Baldi, P.: Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. Bioinformatics **21**(suppl_1) (2005) i359–368
10. Willett, P.: Similarity-based virtual screening using 2D fingerprints. Drug Discovery Today **11**(23/24) (2006) 1046–1051
11. Joachims, T.: Learning to Classify Text using Support Vector Machines: Methods, Theory, and Algorithms. Springer (2002)
12. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, AAAI Press (1998) 43–48

13. Hand, D.J.: Measuring classifier performance: a coherent alternative to the area under the ROC curve. Machine Learning **77**(1) (2009) 103–123
14. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**(Jan) (2006) 1–30
15. Watanabe, S.: Information theoretical analysis of multivariate correlation. IBM Journal of Research and Development **4**(1) (1960) 66–82
16. De Raedt, L.: Logical and Relational Learning. Springer (2008)
17. Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining. In Džeroski, S., Lavrač, N., eds.: Relational Data Mining. Springer-Verlag (2001) 262–291
18. Plotkin, G.: A further note on inductive generalization. In: Machine Intelligence. Volume 6. Edinburgh University Press (1971) 101–124
19. De Raedt, L., Ramon, J.: Deriving distance metrics from generality relations. Pattern Recognition Letters **30**(3) (2009) 187–191
20. Raymond, J., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. Journal of Computer-Aided Molecular Design **16** (2002) 521–533
21. Karunaratne, T., Boström, H.: Learning to classify structured data by graph propositionalization. In: Proceedings of the Second IASTED International Conference on Computational Intelligence. (2006) 393–398
22. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering **17**(8) (2005) 1036–1050
23. Kramer, S., De Raedt, L., Helma, C.: Molecular feature mining in HIV data. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01), ACM Press (2001) 136–143
24. He, H., Singh, A.K.: Graphrank: Statistical modeling and mining of significant subgraphs in the feature space. In: ICDM '06: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society (2006) 885–890
25. Sebag, M.: Distance induction in first order logic. In Lavrač, N., Džeroski, S., eds.: Proceedings of the Seventh International Workshop on Inductive Logic Programming. Volume 1297 of Lecture Notes in Artificial Intelligence., Springer (1997) 264–272
26. Chaoji, V., Al Hasan, M., Salem, S., Besson, J., J. Zaki, M.: Origami: A novel and effective approach for mining representative orthogonal graph patterns. Stat. Anal. Data Min. **1**(2) (2008) 67–84
27. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters **19** (1998) 255–259
28. Gärtner, T.: Kernels for Structured Data. PhD thesis, University of Bonn, Germany (2005)
29. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. (2004) 158–167
30. Ceroni, A., Costa, F., Frasconi, P.: Classification of small molecules by two- and three-dimensional decomposition kernels. Bioinformatics **23**(16) (2007) 2038–2045
31. Ben-David, S., Eiron, N., Simon, H.U.: Limitations of learning via embeddings in euclidean half spaces. J. of Mach. Learning Research **3** (2002) 441–461

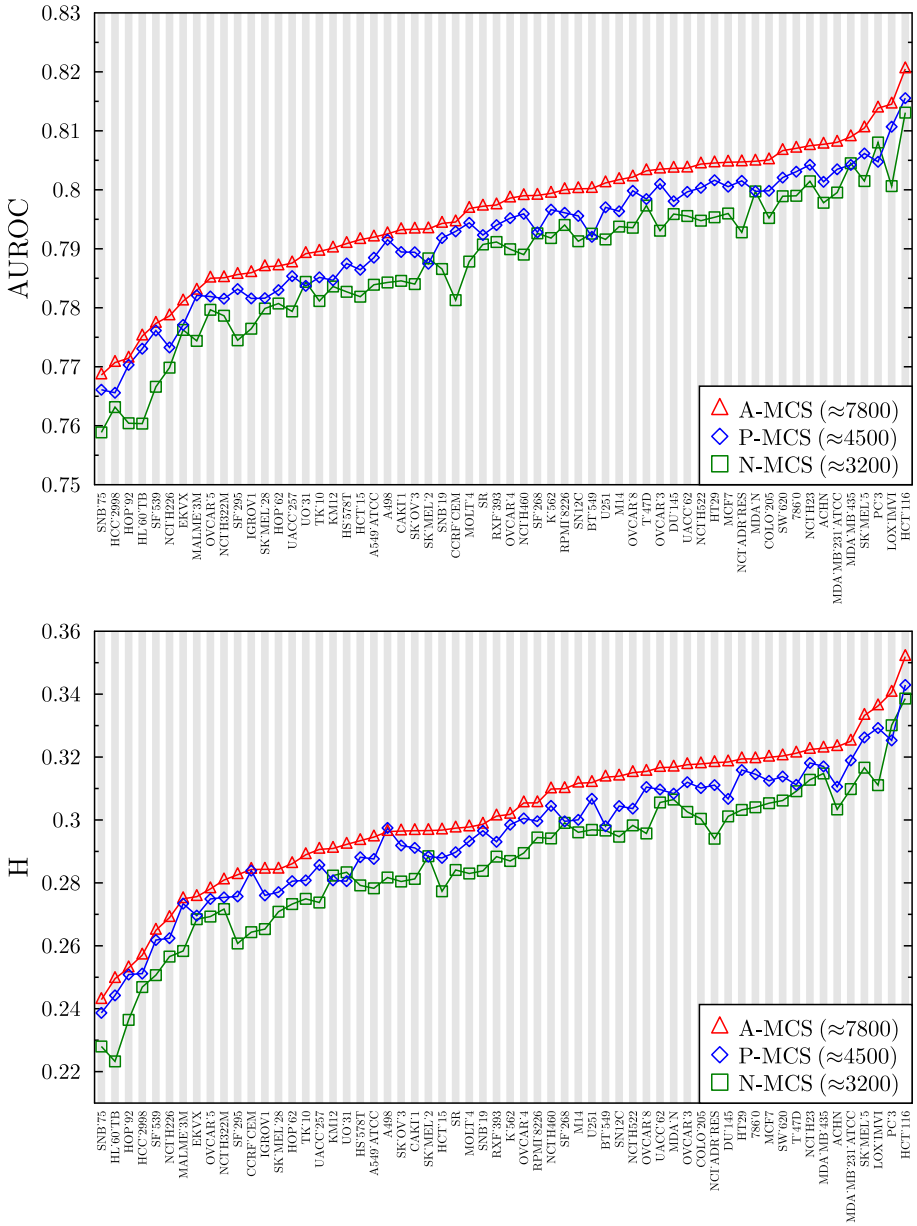

Fig. 3. Predictive performance (AUROC and H) of different selection strategies on 60 NCI datasets.

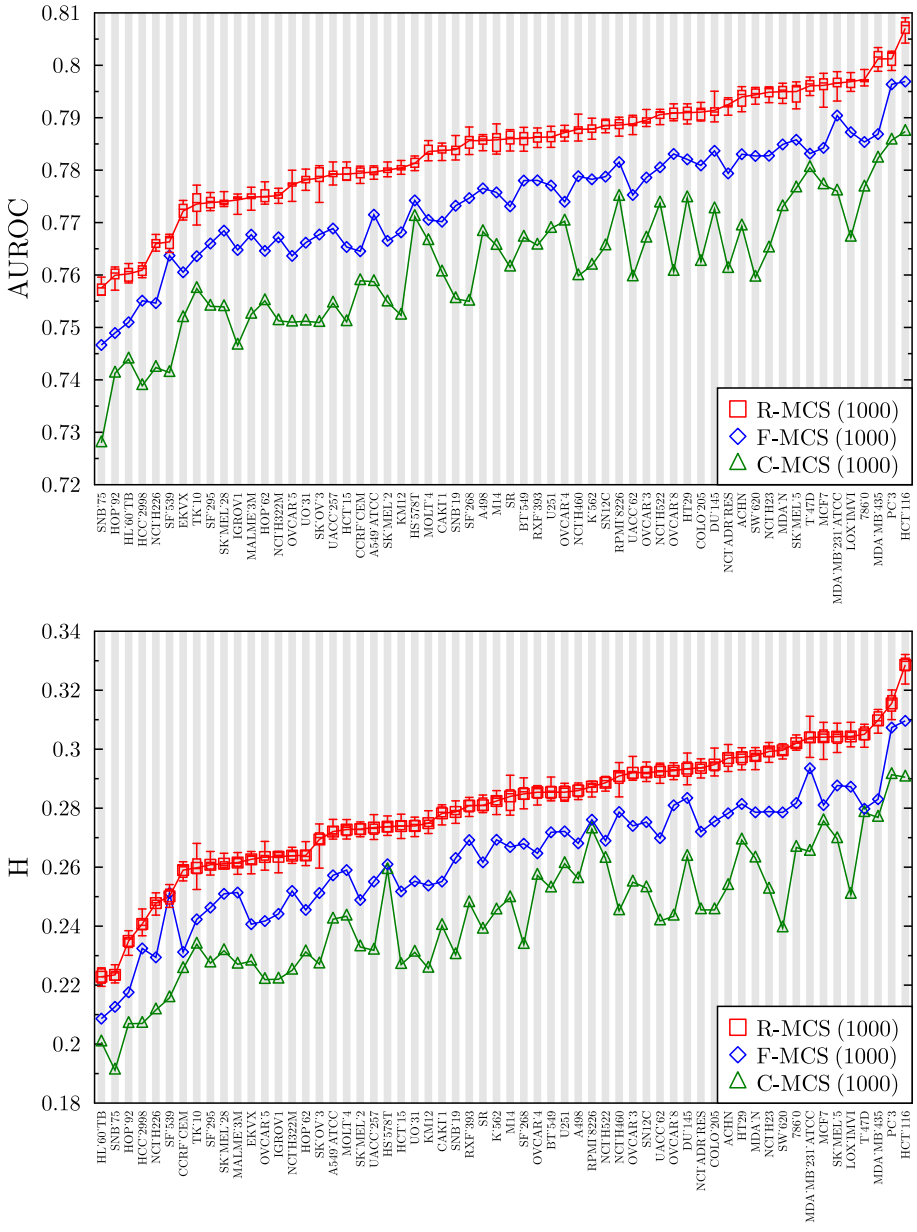

Fig. 4. Predictive performance (AUROC and H) when applying different constraints on 60 NCI datasets.

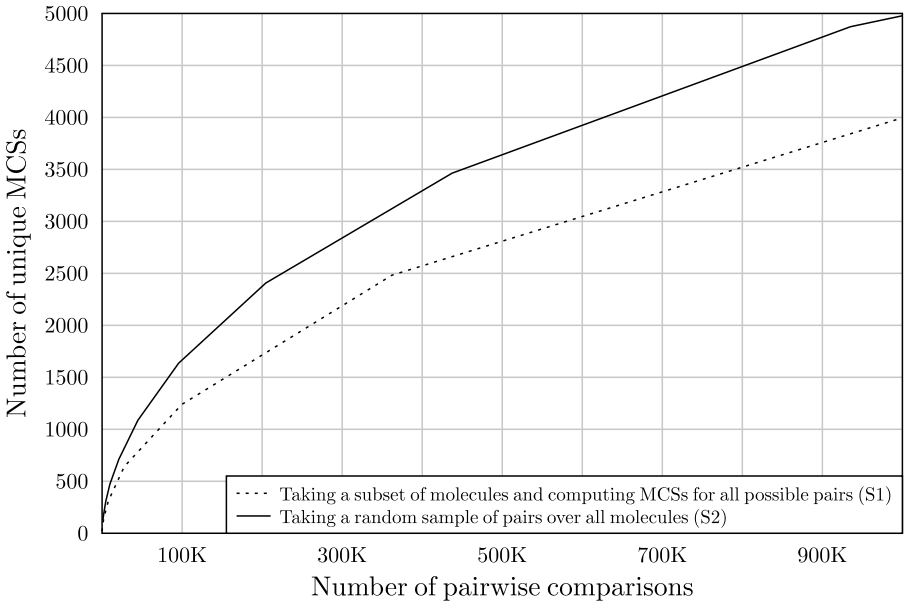

Fig. 5. Relationship between the numbers of pairwise comparisons and the number of unique MCSs.

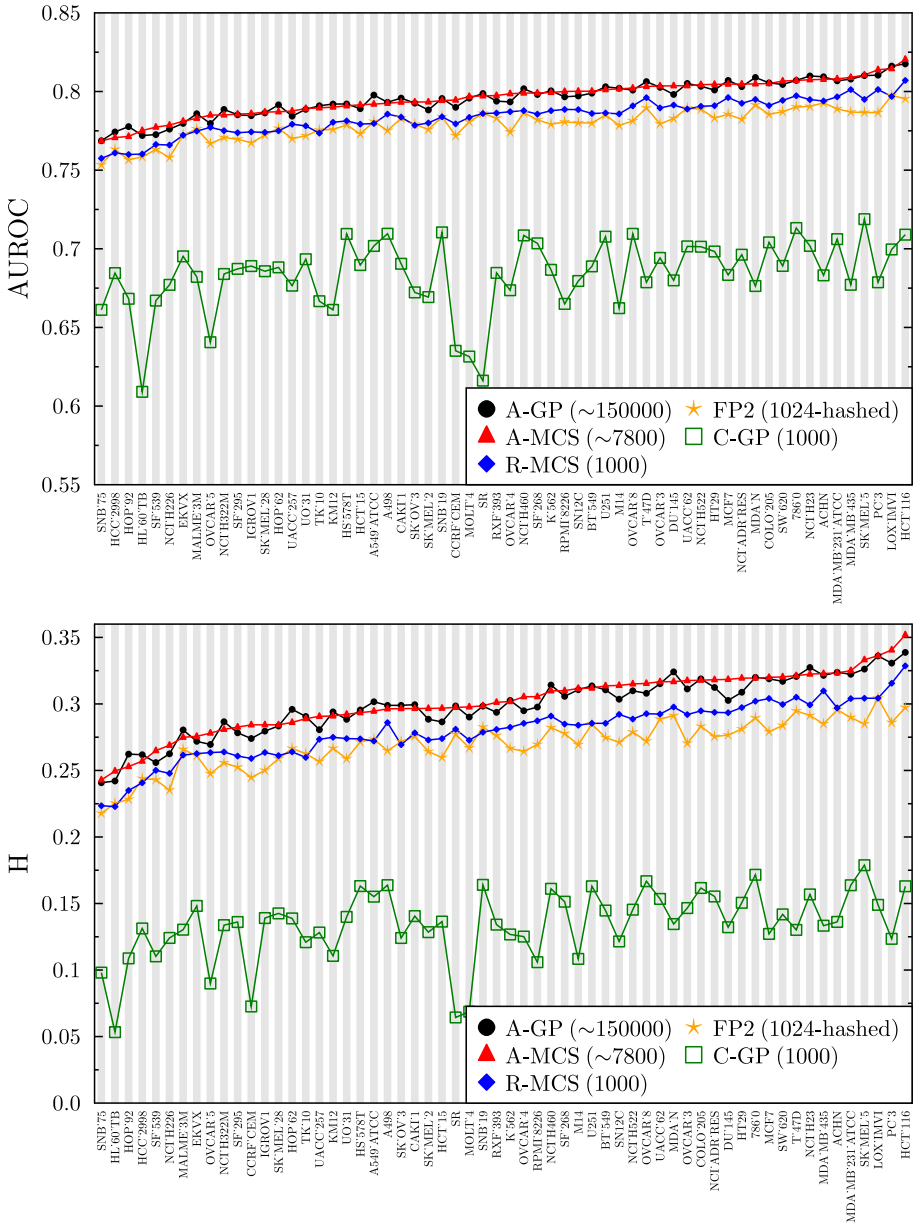

Fig. 6. Predictive performance of state-of-the-art feature generation methods on 60 NCI datasets.